```
# ======================================================================
# TENSORIUM v3.1 - ENFOQUE DE DENSIDAD DE BRAZOS (RESONANCIA DE FASE)
# Modulación exacta de T22 para esculpir los 2 brazos principales
# ======================================================================

import numpy as np
import matplotlib.pyplot as plt
import ipywidgets as widgets
from IPython.display import display, clear_output

# --- REJILLA MATEMÁTICA ESTÁNDAR ---
radios = np.linspace(1.0, 18.0, 50)    # Más resolución radial para nitidez
angulos = np.linspace(0, 2*np.pi, 70)  # Más resolución angular
R, ANGULO = np.meshgrid(radios, angulos)

def renderizar_tensorium_v3_1(change=None):
  """
  Usa T22 para crear una onda de densidad angular. Presiona las líneas
  de corriente en dos brazos definidos sin deformar la frontera circular.
  """
  with output_area:
    clear_output(wait=True)

    try:
      T00 = float(txt_t00.value)
      T03 = float(txt_t03.value)
      Trphi = float(txt_trphi.value)
      Tcolim = float(txt_tcolim.value) # Componente T22
    except ValueError:
      T00, T03, Trphi, Tcolim = 3.0, 4.0, 20.0, 1.5

    # --- ALGORITMO GEOMÉTRICO PURIFICADO ---

    # 1. Componente T00 y Trphi (Aplanamiento y extensión radial)
    factor_masa = np.maximum(0.1, 1.0 - (T00 / (R**2 + Trphi)))
    R_curvado = R * factor_masa

    # 2. Componente T03 (Vórtice base)
    perfil_arrastre = T03 * (4.0 / (R_curvado + 0.5)) + (Trphi * 0.1 * R_curvado)

    # 3. Componente T22 (Compresión Cuadrupolar Angular en la Fase)
    # ESTE ES EL CAMBIO CLAVE: T22 modula la velocidad angular de las líneas.
    # Al sumar un seno de ángulo doble dentro de la torsión, obligamos a las lineas
```

```
    # a agruparse en 2 zonas y espaciarse en las otras, simulando ondas de densidad.
    fase_brazos = Tcolim * np.sin(2.0 * ANGULO - 1.5 * R_curvado)
    ANGULO_VÓRTICE = ANGULO + perfil_arrastre + fase_brazos

    # Proyección cartesiana limpia (mantiene la frontera circular de la galaxia)
    X = R_curvado * np.cos(ANGULO_VÓRTICE)
    Y = R_curvado * np.sin(ANGULO_VÓRTICE)

    # Flexión vertical en Z
    Z_flexion = (0.1 * T00 * np.sin(R_curvado) / (R_curvado + 1.0)) / (1.0 + np.abs(Trphi)*0.1)

    # --- CONFIGURACIÓN GRÁFICA ---
    fig = plt.figure(figsize=(10, 8), facecolor='#0f0f1d')
    ax = fig.add_subplot(111, projection='3d')
    ax.set_facecolor('#0f0f1d')

    # Centro Activo
    color_centro = '#ff3366' if T00 >= 0 else '#00ffcc'
    ax.scatter([0], [0], [0], color=color_centro, s=250, zorder=10)

    # Mapeo de color continuo plasma
    color_map = plt.cm.plasma(1.0 - (R_curvado / 18.0))

    # Renderizado de la superficie
    ax.plot_surface(X, Y, Z_flexion, facecolors=color_map, alpha=0.6,
                    edgecolor='#ffffff', linewidth=0.06, shade=True)

    ax.set_xlim([-20, 20])
    ax.set_ylim([-20, 20])
    ax.set_zlim([-5, 5])
    ax.axis('off')
    ax.view_init(elev=38, azim=-55)

    ax.set_title(f"Tensorium v3.1: Ondas de Densidad Cuadrupolar ($T_{{22}}$)\n"
                 f"$T_{{00}}$ = {T00:.1f} | $T_{{03}}$ = {T03:.1f} | $T_{{r\\phi}}$ = {Trphi:.1f} | $T_{{22}}$
(Brazos) = {Tcolim:.2f}",
                 color='white', fontsize=11, pad=20)

    plt.show()

# --- INTERFAZ ---
txt_t00 = widgets.Text(value='3.0', description='T_00 (Masa):', layout={'width': '180px'})
txt_t03 = widgets.Text(value='4.0', description='T_03 (Vórtice):', layout={'width': '180px'})
txt_trphi = widgets.Text(value='20.0', description='T_rϕ (Viscosidad):', layout={'width': '200px'})
```

```
txt_tcolim = widgets.Text(value='1.2', description='T22 (Brazos):', layout={'width': '200px'})

txt_t00.observe(renderizar_tensorium_v3_1, names='value')
txt_t03.observe(renderizar_tensorium_v3_1, names='value')
txt_trphi.observe(renderizar_tensorium_v3_1, names='value')
txt_tcolim.observe(renderizar_tensorium_v3_1, names='value')

output_area = widgets.Output()
ui_layout = widgets.VBox([widgets.HBox([txt_t00, txt_t03]), widgets.HBox([txt_trphi,
txt_tcolim])])

display(ui_layout)
display(output_area)

renderizar_tensorium_v3_1()
```